XTR Extended to GF(p^{6m})

Seongan Lim[1], Seungjoo Kim[1], Ikkwon Yie[2,⋆],
Jaemoon Kim[2,⋆], and Hongsub Lee[1]

[1] KISA (Korea Information Security Agency),
5th FL., Dong-A Tower, 1321-6, Seocho-Dong, Seocho-Gu, Seoul 137-070, Korea
{seongan, skim, hslee}@kisa.or.kr
[2] Department of Mathematics, Inha University,
YongHyun-Dong, Nam-Gu, Incheon, Korea
{ikyie, jmkim}@math.inha.ac.kr

Abstract. A. K. Lenstra and E. R. Verheul in [2] proposed a very efficient way called XTR in which certain subgroup of the Galois field GF(p^6) can be represented by elements in GF(p^2). At the end of their paper [2], they briefly mentioned on a method of generalizing their idea to the field GF(p^{6m}). In this paper, we give a systematic design of this generalization and discuss about optimal choices for p and m with respect to performances. If we choose m large enough, we can reduce the size of p as small as the word size of common processors. In such a case, this extended XTR is well suited for the processors with optimized arithmetic on integers of word size.

1 Introduction

After Diffie-Hellman (DH) key agreement protocol was published, many related key agreement protocols have been proposed. Very recently, in [2] A. K. Lenstra and E. Verheul proposed an efficient computational tool called XTR (Efficient and Compact Subgroup Trace Representation) and showed that it can be adopted to various public key systems including key exchange protocols. Their scheme results in relatively efficient system with respect to the computational and communicational complexity compared to currently known public key schemes using subgroups. At the end of their paper, they mentioned very briefly that XTR can be generalized in a straightforward way using the extension field of the form GF(p^{6m}) and made some general comments with the focus on the case $p = 2$.

In this paper, we carry out the generalization in detail and discuss about optimal choices of the parameters p and m. The idea is mostly straightforward, but we need to be more systematic to find out optimal choices of p and m among the possible cases. In more detail, the generalization is done in two steps. First, we propose a systematic design for XTR-like system in GF(p^{6m}) using an irreducible cubic polynomial $F(c, X) = X^3 - cX^2 + c^{p^m}X - 1 \in \mathrm{GF}(p^{2m})[X]$ for

⋆ Yie and Kim's work was supported by INHA Univ. Research Grant (INHA-21072).

S. Vaudenay and A. Youssef (Eds.): SAC 2001, LNCS 2259, pp. 301–312, 2001.

any m. Then we determine the required properties of the parameters m, p, and c for $F(c, X)$ under the efficiency and security considerations. We are focusing on the case that is efficient in the limited applications such as smart cards. We use an optimal normal basis to represent elements of $\mathrm{GF}(p^{2m})$ over $\mathrm{GF}(p)$. Hence we consider the case where $2m+1$ is a prime and p is a primitive element modulo $2m+1$. We suggest to use m such that either $2m+1$ is a Fermat prime or both m and $2m+1$ are primes. With such a choice of m, a randomly chosen prime p has a better chance to be a primitive element in Z_{2m+1}.

We estimated the required computational complexity for XTR extended to $\mathrm{GF}(p^{6m})$ under considerations as the above, and compare the result with XTR in $\mathrm{GF}(P^6)$ where P and p^{6m} have the same bit sizes. The result shows us that the required number of bit operations for both are about the same.

Modern workstation microprocessors are designed to calculate in units of data known as words. For large prime p, multiple machine words are required to represent elements of prime field $\mathrm{GF}(p)$ on microprocessors, since typical word sizes are not large enough. This representation causes two possible computational difficulties: carries between words must be treated and reduction modulo p must be performed with operands of multiple span words.

Hence we see that using prime number p as small as the word size of common processors for XTR relieves the above computational difficulties for operation in $\mathrm{GF}(P)$ for large prime P. In this way, the proposed generalization maintains the communicational advantages as in XTR and it enhances the computational advantages over XTR since there is no need of multiprecision computation especially when the system is implemented under workstation processors with optimized arithmetic on integers of word size.

Unfortunately, there are some drawbacks in extending XTR to $\mathrm{GF}(p^{6m})$. As m gets larger there are fewer prime numbers p, q that can be used to establish an XTR-like system in $\mathrm{GF}(p^{6m})$. Also it takes longer to generate the primes p, q with the required properties. But generating p, q is a one-time task and it is not a serious disadvantage in many cases.

Our paper is organized as the following. In Section 2, we describe the generalized system in such a way that it can be formulated for any m, setting aside any security or complexity related concerns. In Section 3, we discuss about the security of the system and determine the choices of parameters. In Section 4, we estimate the computational complexity for the proposed XTR-like scheme in $\mathrm{GF}(p^{6m})$. Then we discuss about optimal choices of parameters. In Section 5, we conclude our paper with comparison of efficiency and security for cryptographic schemes using this generalization with those using the original XTR under the Galois fields with about the same sizes. We also discuss about an efficient way of parameter generation of XTR system extended to $\mathrm{GF}(p^{6m})$ and recommend good choices of m for current use in Appendix A.

2 A Description of XTR Extended to GF(p^{6m})

In this section we describe the XTR system using elementary symmetric polynomials and give a systematic way to generalize XTR-like system to GF(p^{6m}). Following [2], we start by setting up some notations. Given an element $c \in \mathrm{GF}(p^{2m})$, we define the cubic polynomial $F(c, X)$ as following:

$$F(c, X) = X^3 - cX^2 + c^{p^m} X - 1 = (X - h_0)(X - h_1)(X - h_2),$$

where the roots h_i's are taken from the splitting field of $F(c, X)$. We set $c_n = h_0^n + h_1^n + h_2^n$ for any integer n. Then from the root-coefficient relations of a cubic equation, we have

Lemma 1. *For any integers n and t we have*

1. $c_1 = c$, $h_0h_1 + h_1h_2 + h_0h_2 = c^{p^m}$, *and* $h_0h_1h_2 = 1$;
2. $c_{-n} = c_{np^m} = c_n^{p^m}$;
3. *Either all h_i's are in* GF(p^{2m}) *or $F(c, X)$ is irreducible over* GF(p^{2m}) *and all h_i's have order dividing $p^{2m} - p^m + 1$;*
4. $(c_n)_t = c_{tn} = (c_t)_n$.

Proof. Item 1 is nothing but the root-coefficient relation. Items 2 and 3 can be proved exactly the same way as Lemma 2.3.2 of [2] is proved.

To prove item 4, note that

$$c_n = h_0^n + h_1^n + h_2^n,\ c_n^{p^m} = c_{-n} = (h_1h_2)^n + (h_0h_2)^n + (h_0h_1)^n.$$

This implies that

$$F(c_n, X) = X^3 - c_nX^2 + c_n^{p^m} X - 1 = (X - h_0^n)(X - h_1^n)(X - h_2^n).$$

And thus we see that

$$(c_n)_t = (h_0^n)^t + (h_1^n)^t + (h_2^n)^t = h_0^{nt} + h_1^{nt} + h_2^{nt}.$$

Hence we see that $(c_n)_t = c_{nt} = (c_t)_n$ for any integer n, t.

It can be easily checked that any irreducible cubic polynomial $f(x) = x^3 - ax^2 + bx - 1 \in \mathrm{GF}(p^{2m})$ is of the form $f(x) = x^3 - ax^2 + a^{p^m} x - 1$ if the order of the roots $h_0, h_1, h_2 \in \mathrm{GF}(p^{6m})$ of $f(x) = 0$ divides $p^{2m} - p^m + 1$.

Recall that the elementary symmetric polynomials σ_k of degree k in the indeterminates X_1, X_2, X_3 are given by

$$\sigma_1 = X_1 + X_2 + X_3, \quad \sigma_2 = X_1X_2 + X_2X_3 + X_1X_3, \quad \sigma_3 = X_1X_2X_3.$$

Here is a theorem, due to Newton, so-called 'fundamental theorem on symmetric polynomials'.

Theorem 1. *(Theorem 1.75 in [7]) Let $\sigma_1, \sigma_2, \sigma_3$ be the elementary symmetric polynomials in X_1, X_2, X_3 over a commutative ring R, and let $s_0 = 3$, $s_n = X_1^n + X_2^n + X_3^n \in R[X_1, X_2, X_3]$ for $n \geq 1$. Then the following equality*

$$s_k - s_{k-1}\sigma_1 + s_{k-2}\sigma_2 - s_{k-3}\sigma_3 = 0$$

holds for $k \geq 3$.

As a direct application of Newton's Theorem the following lemma can be easily proved.

Lemma 2. *Then for any positive integer n we have*

1. $c_{n+2} = c_{n+1}c - c_n c^{p^m} + c_{n-1}$, $c_{n-1} = c_{n+2} - c_{n+1}c + c_n c^{p^m}$;
2. $c_{2n} = c_n^2 - 2c_n^{p^m}$;
3. $c_{2n+1} = c_n c_{n+1} - c c_n^{p^m} + c_{n-1}^{p^m}$;
4. $c_{2n-1} = c_n c_{n-1} - c^{p^m} c_n^{p^m} + c_{n+1}^{p^m}$.

As in [2], we denote $S_n(c) = (c_{n-1}, c_n, c_{n+1})$ for any integer n. Then by Lemma 2.2, we see that $S_{-1}(c) = (c^{2p^m} - 2c, c^{p^m}, 3)$, $S_0(c) = (c^{p^m}, 3, c)$, and $S_1(c) = (3, c, c^2 - 2c^{p^m})$. Thus, if we do not care about efficiency or security we can define XTR-like key exchange system as following for any prime p and positive integer m and for any $c \in \mathrm{GF}(p^{2m})$.

XTR-DH key exchange system in $\mathbf{GF}(p^{6m})$:

1. Alice chooses a random integer n and computes $S_n(c)$ then sends c_n to Bob.
2. Bob chooses a random integer t and computes $S_t(c)$ then sends c_t to Alice.
3. Alice and Bob share the key c_{nt} that can be obtained by computing either $S_n(c_t) = ((c_t)_{n-1}, (c_t)_n, (c_t)_{n+1})$ or $S_t(c_n) = ((c_n)_{t-1}, (c_n)_t, (c_n)_{t+1})$.

All the XTR-based schemes given in [2] can be extended to $\mathrm{GF}(p^{6m})$ similarly. In the following sections, we will discuss about the parameter selections to meet various security levels and to boost up the efficiency.

3 Parameter Selection for Security Consideration

Various XTR-based public key systems or key exchange protocols rely their security on the Discrete Logarithm Problem(DLP) in the base $g \in \mathrm{GF}(p^{6m})$, where g is a root of the cubic equation $F(c, X) = 0$. Therefore, in order to make XTR-based schemes secure, we need to use parameters p, m, c such that the DLP in the base g is difficult. In this section, we follow the method in [1] to determine the size of the subgroup generated by g to prevent known attacks on DLP in extension fields.

Up to this point one of the best attacks known for the DLP is the Index Calculus Method using Number Field Sieve. For a DLP in the base $g \in \mathrm{GF}(p^{6m})$,

the asymptotic complexity of the Index Calculus Method using Number Field Sieve is

$$L(p^s, 1/3, 1.923 + o(1)),$$

where s is the smallest divisor of $6m$ such that g is contained in a subfield of $GF(p^{6m})$ isomorphic to $GF(p^s)$. Thus, for the security reason, it is desirable to use $g \in \mathrm{GF}(p^{6m})$, which is not contained in any proper subfield of $\mathrm{GF}(p^{6m})$. Note that if a root g of the polynomial $F(c, X)$ is not contained any proper subfield of $\mathrm{GF}(p^{6m})$ then $F(c, X)$ is irreducible over $\mathrm{GF}(p^{2m})$. Hence we have $c = \mathrm{Tr}(g)$ and the roots of $F(c, X)$ are conjugates of g over $\mathrm{GF}(p^{2m})$. Here, $\mathrm{Tr} : \mathrm{GF}(p^{6m}) \to \mathrm{GF}(p^{2m})$ is the trace projection defined by $\mathrm{Tr}(x) = x + x^{p^{2m}} + x^{p^{4m}}$ for $x \in \mathrm{GF}(p^{6m})$.

The following Lemma, which follows directly from Lemma 2.4 of [1], gives a sufficient condition for a subgroup of $\mathrm{GF}(p^{6m})^*$ not to be contained in any proper subfield of $\mathrm{GF}(p^{6m})$.

Lemma 3. *Let q be a prime factor of $\Phi_{6m}(p)$. Then the subgroup of $\mathrm{GF}(p^{6m})^*$ of order q is not contained in any proper subfield of $\mathrm{GF}(p^{6m})$.*

Here $\Phi_n(X)$ is the n-th cyclotomic polynomial for a positive integer n not divisible by p. For computations with cyclotomic polynomials, we can use Theorem 3.27 in [7]:

$$\Phi_{6m} = \prod_{d|6m} (X^{6m/d} - 1)^{\mu(d)},$$

where $\mu(\cdot)$ is the Möbius function.

As we shall see in the next section, we will take m to be a prime or a power of 2 (so that $2m + 1$ is a Fermat prime) for easy selection of the prime p. Thus we have

$$\Phi_{6m}(X) = \frac{(X^{6m} - 1)(X^2 - 1)(X^3 - 1)(X^m - 1)}{(X^{2m} - 1)(X^{3m} - 1)(X^6 - 1)(X - 1)} = \frac{X^{2m} - X^m + 1}{X^2 - X + 1}$$

if m is a prime, or

$$\Phi_{6m}(X) = \frac{(X^{6m} - 1)(X^m - 1)}{(X^{2m} - 1)(X^{3m} - 1)} = X^{2m} - X^m + 1$$

if m is a power of 2.

When we use a system based on the DLP in a multiplicative subgroup of size q of the Galois field, the sizes of q and the underlying Galois field that guarantee the security required currently can be determined according to the table in [4]. The recommended size for q is much larger than $p^2 - p + 1$ for small or medium sized p. Thus in our case we need to take the size q of the subgroup of $\mathrm{GF}(p^{6m})^*$ to be much larger than $p^2 - p + 1$. In addition, if we take q to be a prime factor of $p^{2m} - p^m + 1$ then the calculation in the previous paragraph tells us that q is a prime factor of $\Phi_{6m}(p)$. Then by Lemma 3 the subgroup of order q is not contained in any proper subfield of $\mathrm{GF}(p^{6m})$.

Also it is well-known that birthday-type attacks can be applied to get x from the given g^x and the complexity of birthday attacks can be estimated by $O(\sqrt{q})$ where q is the order of g. Currently the recommended size for q is $q > 2^{160}$. Thus to make our system as secure as the DLP problem in $GF(p^{6m})$ against currently known attacks, it is enough to choose $g \in \mathrm{GF}(p^{6m})$ so that the order of g is a prime factor of $p^{2m} - p^m + 1$ and larger than $\max(p^2 - p + 1, 2^{160})$.

4 Parameter Selection for Efficiency Cconsiderations

The most basic computation required in XTR-like schemes is to compute $S_n(c)$ from any given $c \in \mathrm{GF}(p^{2m})$ and a positive integer n. ¿From Lemma 2 we have

Lemma 4. *Let c be any element of $\mathrm{GF}(p^{2m})$ and $A(c)$ be the 3×3 matrix given by*

$$A(c) = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & -c^{p^m} \\ 0 & 1 & c \end{pmatrix}.$$

For any integer $n \geq 1$, we have $S_{n+1}(c) = S_n(c)A(c)$.

By modifying the 'square and multiply' method, we can compute $S_n(c)$ as follows:

Algorithm 2 *Let $c \in GF(p^{2m})$ and a positive integer n be given. ¿From the binary expansion of $n = \sum_{i=0}^{k} m_i 2^i$ define a sequence $\{t_i\}$ by*

$$\begin{aligned} t_0 &= m_0 \\ t_i &= 2t_{i-1} + m_i, \ i \geq 1. \end{aligned}$$

To compute $S_n(c) = (c_{n-1}, c_n, c_{n+1})$, follow the steps:

Step 1. *Set $S_{t_0}(c) = (c^{p^m}, 3, c)$ if $t_0 = 0$, or $S_{t_0}(c) = (3, c, c^2 - 2c^{p^m})$ if $t_0 = 1$.*
Step 2. *Compute S_{t_i} from $S_{t_{i-1}}$ for $i = 1, 2, \cdots, k$.*
Step 3. *Output S_{t_k}.*

Step 2 of the above algorithm is performed as following. For a fixed i let us let $d = t_{i-1}$ for simplicity. If $m_i = 0$, then

$$S_{t_i} = S_{2d} = (c_d c_{d+1} - c^{p^m} c_d^{p^m} + c_{d+1}^{p^m}, c_d^2 - 2c_d^{p^m}, c_d c_{d+1} - c c_d^{p^m} + c_{d-1}^{p^m}).$$

If $m_i = 1$, then

$$S_{t_i} = S_{2d+1} = (c_d^2 - 2c_d^{p^m}, c_d c_{d+1} - c c_d^{p^m} + c_{d-1}^{p^m}, c_{d+1}^2 - 2c_{d+1}^{p^m}).$$

As we can see from Lemma 2 and Algorithm 2, the most frequently performed operations in our system are the following three types:

$$x^2, xy, xz - yz^{p^m} \quad \text{for } x, y, z \in \mathrm{GF}(p^{2m}).$$

In order to make the XTR system in GF(p^{6m}) efficient, it is enough to perform these operations efficiently. Thus, we consider the case when GF(p^{2m}) has an optimal normal basis (ONB) of type I, that is, the case when $2m+1$ is a prime number and $p \pmod{2m+1}$ is a primitive element in Z_{2m+1}. In this case, the cyclotomic polynomial

$$\Phi_{2m+1}(X) = \frac{X^{2m+1}-1}{X-1} = X^{2m} + X^{2m-1} + \cdots + X + 1 \in \mathrm{GF}(p)[X]$$

is irreducible and the set $B_1 = \{\alpha, \alpha^p, \alpha^{p^2}, \ldots, \alpha^{p^{2m-1}}\}$ of roots becomes an Optimal Normal Basis for GF(p^{2m}). Since $p \pmod{2m+1}$ is a primitive element in Z_{2m+1}, this basis B_1 is setwise equal to the (almost) polynomial basis $B_2 = \{\alpha, \alpha^2, \alpha^3, \ldots, \alpha^{2m}\}$.

The complexity for elementary operations in GF(p^{2m}) is well studied by Lenstra in [1]. Our focus is on the most frequently performed operations

$$x^2, xy, xz - yz^{p^m} \quad \text{for } x, y, z \in \mathrm{GF}(p^{2m})$$

as was in [2]. We have similar result as in [2] on the complexity for these operations.

Lemma 5. *Let p and $2m+1$ be prime numbers, where $p \pmod{2m+1}$ is a primitive element in Z_{2m+1}. Then for $x, y, z \in \mathrm{GF}(p^{2m})$, we have*

1. *Computing x^{p^m} is for free.*
2. *Computing x^2 takes 80 percent of the complexity taken for multiplications in* GF(p^{2m}).
3. *Computing xy takes $4m^2$ multiplications in* GF(p).
4. *Computing $xz - yz^{p^m}$ takes $4m^2$ multiplications in* GF(p).

Proof. Since we can use either of the two bases B_1 and B_2 at our convenience, all the items are straightforward. Thus we only prove item 4 in detail. Set $t = 2m$. First we represent x, y, z by using the normal basis B_1,

$$x = \sum_{i=0}^{t-1} a_i \alpha^{p^i}, y = \sum_{i=0}^{t-1} b_i \alpha^{p^i} z = \sum_{i=0}^{t-1} c_i \alpha^{p^i},$$

and we get

$$z^{p^m} = \sum_{i=0}^{m-1} c_i \alpha^{p^{i+m}} + \sum_{i=m}^{t-1} c_i \alpha^{p^{i-m}}.$$

Then we have

$$xz - yz^{p^m} = \sum_{i=0}^{t-1} \sum_{j=0}^{m-1} a_i c'_j \alpha^{p^i} \alpha^{p^{j+m}} + \sum_{i=0}^{t-1} \sum_{j=m}^{t-1} b_i c''_j \alpha^{p^i} \alpha^{p^{j-m}},$$

where $c'_j = c_j + c_{m+j}$ and $c''_j = c_j + c_{j-m}$. Getting c'_j, c''_j from c_j's is a free operation. Now we convert the basis into B_2 and then the computation for

$\alpha^{p^j}\alpha^{p^{i\pm m}}$ becomes free. Hence we need t^2 multiplications in GF(p) to compute $xy - yz^{p^m}$.

We note here that we haven't used any high speed multiplication algorithms in the proof. When we apply more efficient multiplication algorithms, then the required bit complexity will be reduced. Comparing Lemma 2.1.1 in [2] and the above Lemma 5, we see that the number of bit operations for computing

$$x^2, xy, xz - yz^p$$

are about the same in the following two cases when $|P| = m|p|$,

- $x, y, z \in \mathrm{GF}(P^2)$ with $P = 2 \pmod 3$;
- $x, y, z \in \mathrm{GF}(p^{2m})$ where $2m + 1$ is a prime and $\langle p \rangle = Z^*_{2m+1}$.

Now based on Lemma 5, we have the following estimate for the computational complexity of Algorithm 2.

Theorem 3. *Let $c \in \mathrm{GF}(p^{2m})$ and a positive integer n be given. Then it takes at most $11.2m^2 \log n$ multiplications in $\mathrm{GF}(p)$ to compute $S_n(c) = (c_{n-1}, c_n, c_{n+1})$.*

Also we estimate the required bit complexity to compute $\mathrm{Tr}(g^a g^{bk})$ from given $\mathrm{Tr}(g)$ and $S_k(\mathrm{Tr}(g))$ for unknown k. This computation is required for XTR-Nyberg-Rueppel signature scheme as in [2].

Theorem 4. *Let $g \in \mathrm{GF}(p^{6m})$ be an element of prime order q and suppose $Tr(g)$ and $S_k(Tr(g))$ be given for some unknown positive integer k. Let a, b be positive integers with less than q. Then $Tr(g^a g^{bk})$ can be computed at a cost of $(11.2 \log(a/b \pmod q) + 11.2 \log b + 36)m^2$ multiplications in GF(p).*

We specify the steps as following :

- Compute $e = \frac{a}{b} \pmod q$.
- Compute $Tr(g^{k+e})$.
- Compute $Tr(g^{(k+e)b}) = Tr(g^a g^{bk})$

We focus on the second item here. We have

$$S_k(Tr(g))A(c)^e = [S_0(Tr(g))A(c)^k]A(c)^e = [S_0(Tr(g))A(c)^e]A(c)^k$$

In order to get $Tr(g^{k+e})$, what we need is $[S_0(Tr(g))A(c)^e]C(A(c)^k)$, where $C(A(c)^k)$ is the center column of the matrix $A(c)^k$. Since we have already given $S_k(Tr(g))$, and we know that

$$S_k(Tr(g))^T = \begin{pmatrix} S_{-1}(c) \\ S_0(c) \\ S_1(c) \end{pmatrix} C(A(c)^k),$$

where $S_k(Tr(g))^T$ is the transpose of $S_k(Tr(g))$. Hence we have the very same formula as in XTR,

$$C(A(c)^k) = \begin{pmatrix} S_{-1}(c) \\ S_0(c) \\ S_1(c) \end{pmatrix}^{-1} S_k(Tr(g))^T.$$

This implies that computing $C(A(c)^k)$ takes constant time for any k when $S_k(Tr(g))$ is given. In fact, it takes at most $9 \times 4m^2 = 36m^2$ multiplications in GF(p). Hence the complexity to compute $Tr(g^{k+e})$ can be estimated as $(11.2 \log e + 36)m^2$ multiplications in GF(p). And the complexity to compute the third item is $11.2 \log bm^2$ multiplications in GF(p). Thus we can estimate the complexity to get $Tr(g^a g^{bk})$ by $(11.2 \log(a/b \pmod q) + 11.2 \log b + 36)m^2$ multiplications in GF(p).

Thus we see that the computational complexity is about the same for the original XTR and our extension to $GF(p^{6m})$. But the multiprecision problem that occurs in the operations in GF(P) for large P can be removed when we use GF(p) with p as small as the word size of the processor.

Now we pose another condition on m so that it is easy to generate the prime p. Since we are using ONB, p is necessarily a primitive element in Z_{2m+1}. But as a matter of parameter generation, we will decide m first and choose the prime p somewhat randomly. So it is desirable that Z_{2m+1} has more primitive elements.

The number of primitive elements in Z_{2m+1} is $\phi(\phi(2m+1)) = \phi(2m)$, where $\phi(\cdot)$ is the Euler totient function. Thus we want to make $\phi(2m)$ as big as possible compared to m. There are two possible directions we can take to this end. The first is to take m to be a power of 2. But then $2m + 1$ must be a Fermat prime, which is very rare. The other is to take m to be a prime. There are reasonably many choice of primes m for which $2m + 1$ is also a prime.

For appropriate selection of m for current recommendation is given in the Appendix A. After m and the sizes of p, q have been established, we generate p, q and $c = \mathrm{Tr}(g)$. Algorithms for generating parameters are also given in Appendix A.

5 Conclusion

Thus most of the details in XTR can be generalized systematically to the finite field GF(p^{6m}) using the trace projection $\mathrm{Tr} : \mathrm{GF}(p^{6m}) \to \mathrm{GF}(p^{2m})$. Hence it is straightforward to see that the schemes as XTR-DH, XTR-ElGamal encryption, and XTR-Nyberg-Rueppel signatures can be extended to GF(p^{6m}).

For security concerns, all the details were given in Section 5 of [2]. They've discussed about the DLP in GF(p^t), and hence it can be applied to cases in GF(p^{6m}). The communicational and computational advantages of the XTR schemes can be obtained in the generalization as long as we choose m so that either $2m + 1$ is a Fermat prime or both $m, 2m + 1$ are primes and we don't have any multiprecision operations if we select the size of p as small as the word

size of common processors in the generalization but we might need longer time to generate prime numbers p and q in such cases. But the prime numbers are needed to generate only once, the generalized version of XTR is more preferable for limited applications as smart cards.

Acknowledgment

We appreciate the anonymous referees of the SAC 2001 for their comments and suggestions.

References

1. Arjen K. Lenstra, *Using Cyclotomic Polynomials to Construct Efficient Discrete Logarithm Cryptosystems over Finite Fields*, ACISP'97 (1997), **LNCS 1270**, pp. 127–138.
2. Arjen K. Lenstra, Eric R. Verheul, *The XTR public key system*, Advances in Cryptology – CRYPTO'00 **LNCS 1880** (2000), pp. 1–19
3. Arjen K. Lenstra, Eric R. Verheul, *Key improvements to XTR*, Advances in Cryptology – Asiacrypt'00 **LNCS 1976** (2000), pp. 220–233
4. A.K. Lenstra, E.R. Verheul, *Selecting Cryptographic Key Sizes*, http://www.cryptosavvy.com (1999).
5. Arjen K. Lenstra, Eric R. Verheul, *Fast irreduciblility and subgroup membership testing in XTR*, Proceedings of the PKC'01 **LNCS 1992** (2001), pp. 73–86
6. A.E.Brouwer, R.Pellikaan, E.R. Verheul, *Doing More with Fewer Bits*, Advances in Cryptology – Asiacrypt'99, **LNCS 1716** (1999), pp. 321–332.
7. Rudolf Lidl, Harald Niederreiter, *Introduction to finite fields and their applications*, Cambridge, 1994.

A An Efficient Way for Parameter Generation

In this appendix, we describe an efficient way for parameter generation one by one. At first, we decide the size of the field $\mathrm{GF}(p^{6m})$ and which m to use. Then we select the primes p, q. And finally, we select $c \in \mathrm{GF}(p^{2m}) \setminus \mathrm{GF}(p^m)$.

As we discussed in Section 4, m will be chosen so that either $2m+1$ is a Fermat prime or both m and $2m+1$ are primes. In the Table 1 below, we give a list of m which meet this criterion. The numbers in the column titled 'ratio' are the proportion of primitive elements of Z_{2m+1}. The case $m = 1$ is the original XTR. Not much improvement is achieved in the cases $m = 2, 3$. Thus we do not recommend to use these cases. Note that the ratio tends to $1/2$ as m gets larger.

For given m as in the above table, we choose the appropriate size for the field characteristic p so that the size of p^{6m} is about the same as the recommended size for prime fields with respect to the security concerns in the DLP in prime fields. For example, see Table 2.

Now we consider the generation of the prime numbers p, q in our scheme. We follow the scheme of generating prime numbers as in [1] rather than using the

Table 1. Choices for m and the corresponding extension fields

m	ratio	field extensions
1	1/3	GF(p) → GF(p^2) → GF(p^6)
2	2/5	GF(p) → GF(p^4) → GF(p^{12})
3	2/7	GF(p) → GF(p^6) → GF(p^{18})
5	4/11	GF(p) → GF(p^{10}) → GF(p^{30})
8	8/17	GF(p) → GF(p^{16}) → GF(p^{48})
11	10/23	GF(p) → GF(p^{22}) → GF(p^{66})
23	22/47	GF(p) → GF(p^{46}) → GF(p^{138})
29	28/59	GF(p) → GF(p^{58}) → GF(p^{174})
41	40/83	GF(p) → GF(p^{80}) → GF(p^{240})

Table 2. Choices for m and the corresponding size of a finite field

field size	1024 bit	2048 bit	2700 bit	5100 bit
recommended m for p of 16 bits	11	23	29	41
recommended m for p of 32 bits	8	11	23	29
recommended m for p of 64 bits	5	8	11	23

method given in [2]. In general setting, we are interested in the cases using small or medium sized prime number p covered in [1].

Here we describe the method of generating prime numbers p and q that we need. First we determine the $|p^{6m}|$ and $|q|$ (the bit sizes of p^{6m} and q) for general security considerations according to [4]. And then we decide m so that p is of the word size of the processor to be used.

We repeat selecting p until $p^{2m} - p^m + 1$ has a prime factor of size $|q|$. We refer the result of [1] that points it works sufficiently quickly in practice. As m grows, hence p gets smaller, there are fewer appropriate p, m's than XTR case (assuming comparable levels of security). The exact distribution of such primes p, q is not known until now. Since this is one-time cost, it's not a serious disadvantage.

Now consider the generation of $c = Tr(g)$. The most elementary way to generate $c = Tr(g)$ where $g \in GF(p^{6m})$ of the order q with q divides $p^{2m} - p^m + 1$. As usual, first we randomly generate $h \in GF(p^{6m})$ and check if $h^{\frac{p^{2m}-p^m+1}{q}} \neq 1$ and set $g = h^{\frac{p^{2m}-p^m+1}{q}}$. Then such g has the order q. We compute $Tr(g)$. But here we have lemmas which will come in handy when we construct a suitable generator g of a subgroup. And the proofs of these lemmas are similar to the XTR in $GF(P^6)$.

Lemma 6. *Let m be a positive integer such that either $2m+1$ is a Fermat prime or both m and $2m+1$ are primes. Suppose $F(c, X)$ is irreducible over $\mathrm{GF}(p^{2m})$ and $g \in \mathrm{GF}(p^{6m})$ is a root of $F(c, X)$. Then we have $c = \mathrm{Tr}(g)$ and $c_n = \mathrm{Tr}(g^n)$ and the multiplicative order q of g divides $p^{2m} - p^m + 1$.*

Lemma 7. *$F(c, X)$ is reducible over $GF(p^{2m})$ if and only if $c_{p^m+1} \in GF(p^m)$*

By using this lemma, we have a similar algorithm as in XTR to generate $c = Tr(g)$ where $g \in GF(p^{6m})$ of prime order q that is not contained any subfield of $GF(p^{6m})$.

Algorithm to generate $c = Tr(g)$ for our purpose

1. Choose $\tilde{c} \in GF(p^{2m}) \setminus GF(p^m)$.
2. Check if $\tilde{c}_{p^m+1} \in GF(p^m)$. If it is, go to 1.
3. Compute $\tilde{c}_{\frac{p^{2m}-p^m+1}{q}}$ and check if it is 3. If it is not 3, then set $c = \tilde{c}_{\frac{p^{2m}-p^m+1}{q}}$. This c is what we wanted.